

Cut

Consider the following Prolog program.

```
pred(a) .  
pred(x) :- ! .  
pred(b) .
```

Which of the following five queries evaluates to **true**? Please check those and only those queries!

☐ pred(Z), Z=x. ☐ pred(something). ☐ pred(Z), Z=b. ☐ pred(a) ☐ X=b, pred(X).

Recursive Functions

Provide a Prolog predicate **pred/3** that allows for computing the function $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$:

$$f(y, b) = \begin{cases} y^b & \text{if } y = 0 \text{ or } b = 0 \\ b + f(\max(y - 4, 0), b - 1) & \text{else} \end{cases}$$

Example:
?- pred(42,1337,R).

R =

Make sure that:

- the implementation checks that y and b are natural numbers and
- the program provides exactly one answer and terminates afterwards.

Lists

Given a finite list L of elements, provide a simple Prolog predicate **erasedups/3** that allows for computing

- the list L' which agrees with L , but does not contain duplicate entries and
- the number of list elements that have been ignored

Example:

?- erasedups([a,b,c,a,c,d,b,a,x],R,N)

R = [a, b, c, d, x], N = 4.

Provide an implementation of **erasedups/3** without using an accumulator.

Accumulation

Given a finite list L of elements, provide a simple Prolog predicate **makeset/2** that allows for computing

- the list L' which agrees with L , but does not contain duplicate entries

Example:

?- makeset([a,b,c,a,c,d,b,a,x],R)

R = [a, b, c, d, x].

This time, provide an implementation of **makeset/2** that makes use of accumulation and tail recursion.

Given is the following Prolog program, the task is to establish memoization (or caching) for **own_function/1** to make sure that **own_function((B,G))** is computed only once for a given pair **(B,G)**.

```
own_function(( 0,13)) :- !.  
own_function(( 1,72)) :- !.  
  
own_function((B,G)) :-  
    integer(B), var(G),  
    Z1 is random(B), own_function(( Z1 , Y1 )),  
    Z2 is random(B), own_function(( Z2 , Y2 )),  
    Y is 40* Y1 // (Y2+1).
```

Take the above program (via copy-and-paste), introduce a dynamic predicate **memory/1** and make sure that

- the memory is erased on the outer-most call (before the recursive calls)
- all result pairs **(B,G)** are added to the memory at the first position
- the computation of **own_function((B,G))** is only done if **(B,G)** is not yet in the memory

Given a finite directed graph $G = (V, E)$ encoded into the Prolog predicate **edge/2** such that **edge(X,Y)** $\equiv$ **true** iff $(X, Y) \in E$. Provide a predicate **prf/2** that allows for computing the set of nodes $N \subseteq V$, such that

- N contains all nodes that have at least one incoming or one outgoing edge.

Make sure that the resulting list is indeed a set.

Graphs2

No answer

Given a finite directed graph $G = (V, E)$ encoded into the Prolog predicate **edge/2** such that **edge(X,Y)** $\equiv$ **true** iff $(X, Y) \in E$. Provide a predicate **dfs/2** that for a given node **X** ($X \in V$) returns lists of reachable nodes in a depth-first order.